HTTP 1.0 Logs Considered Harmful

Ramón Cáceres, Balachander Krishnamurthy, and Jennifer Rexford

AT&T Labs–Research; 180 Park Avenue
Florham Park, NJ 07932 USA
{*ramon, bala, jrex*}*@research.att.com*

1 Motivation

Virtually all Web performance evaluation work has focused on server logs, proxy logs, or packet traces based on HTTP 1.0 traffic. HTTP 1.1 [1] introduces several new features that may substantially change the characteristics of Web traffic in the coming years. However, there is very little end-to-end HTTP 1.1 traffic in the Internet today. This has led to a dependence on HTTP 1.0 logs and synthetic load generators to postulate improvements to HTTP 1.1, and to evaluate new proxy and server policies. We believe that Web performance studies should use more realistic logs that take into account changes to the HTTP protocol. In particular, we suggest techniques for converting an HTTP 1.0 log into a semi-synthetic HTTP 1.1 log, based on information extracted from packet-level traces and our knowledge of the HTTP 1.1 protocol. As part of this study, we plan to collect detailed packet-level server traces at AT&T's Easy World Wide Web (EW3) platform [2], the Web-hosting part of AT&T WorldNet.

2 Differences Between HTTP 1.0 and HTTP 1.1

The changes in the HTTP protocol address a number of key areas, including caching, hierarchical proxies, persistent TCP connections, and virtual hosts. We focus on specific new features that are likely to alter the workload characteristics (as summarized in Table 1):

- **Persistent connections and pipelining:** Studies of HTTP 1.0 traffic have shown that individual HTTP transfers are typically short-lived, making the round-trip-time for TCP connection establishment a significant portion of the total latency [3, 4]. In HTTP 1.1, HTTP connections are persistent by default, to avoid the round-trip delays, as well as the processing and bandwidth overheads of repeatedly opening and closing connections. The move to persistent connections should result in appreciable differences in the arrival times of requests from the same client (or proxy), particularly for embedded entities. In addition, the spacing of requests would be affected by the (optional) pipelining of successive HTTP requests.

- **Support for semantically transparent caching:** HTTP 1.1 includes more support for caching. Use of the new Expires header, which indicates when a response is considered to be stale, will reduce the need for GET If-Modified-Since requests. HTTP 1.1 also permits response headers to include a unique entity tag for each version of a resource, which can be used as a opaque cache validator for reliable validation. The Cache-Control general header is an explicit

HTTP 1.1 Feature	Implication
Persistent connections	Lowers number of connection set-ups
Pipelining	Shortens interarrival of requests
Expires	Lowers number of validations
Entity tags	Lowers frequency of validations
Max-age, max-stale, etc.	Changes frequency of validations
Range request	Lowers bytes transferred
Chunked encoding	Lowers user perceived latency
Expect/Continue	Lowers error response/bandwidth
Host header	Reduces proliferation of IP addresses

Table 1: Effects of changes in the HTTP protocol

directive mechanism that permits clients or servers to override default actions. For example, max-stale=3600 can be used to extend the time before validation is needed. Proper deployment of caches, and the overall semantic transparency in caching, in HTTP 1.1 should reduce the number of both GET If-Modified-Since requests and 304 Not-Modified responses.

- **Range requests:** HTTP 1.1 range requests allow a client or proxy to request specific subsets of the bytes in a Web resource, instead of the full content. This is useful when only a small portion of the resource is of interest, or to retrieve the remainder of a resource after a partial transfer (e.g., an aborted response). Proxy servers could properly cache ranges to generate range responses later from caches. Origin servers and proxies can reduce the amount of bytes transmitted and thus reduce latency. Range requests will alter the size distribution in the traffic mix and lower user-perceived latency.

- **Chunked encoding:** Chunked encoding facilitates the efficient and secure transmission of dynamically-generated content. Also, by breaking the response into chunks, the server can perform computations on a dynamically generated response (e.g. checksum) without forcing the client to wait. Chunked encoding is likely to lower user-perceived latency of the early part of response, but will not significantly impact the total latency for the response.

- **Expect/Continue:** The Expect request header was added to let a client query a server before sending the rest of the request header. The server must either send a 100 (Continue) response (if it could fulfill the expectation or respond with a 417 (Expectation Failed) error status. If a large PUT could be avoided by receiving the error status bandwidth would be saved. This would reduce load on both the client and the server. It might also lower the number of other error responses from the server as a result of its inability to handle the client's content-body.

- **Multi-homed servers:** Commercial server maintainers wanted to run a single server on a single port but respond with different top-level pages based on the hostname field. Host header will significantly reduce overloading of web servers and reduce the proliferation of IP addresses. This feature is unlikely to impact traffic mix.

3 Using Packet Traces to Model HTTP 1.1

Research on Internet workload characterization has typically focused on creating generative models based on packet traces of various applications [5, 6]. These models range from capturing basic traffic properties, like interarrival and duration distributions, to representing application-level characteristics. Synthetic workload generators based on these models can drive a wide range of simulation experiments, allowing researchers to perform accurate experiments without incurring the extensive overhead of packet trace collection. A synthetic modeling approach has also been applied to develop workload generators for Web traffic [7, 8]. Although these synthetic models of HTTP 1.0 traffic are clearly valuable, it may be difficult to project how these synthetic workloads would change under the new features in HTTP 1.1.

In contrast to Internet packet traces, many sites do maintain Web proxy or server logs. Having a way to convert these HTTP 1.0 logs to representative HTTP 1.1 logs would allow these sites to evaluate the potential impact of various changes to the protocol. These semi-synthetic HTTP 1.1 traces could also be converted into synthetic workload models that capture the characteristics of HTTP 1.1. The process of converting HTTP 1.0 logs to representative HTTP 1.1 logs requires insight into the components of delay in responding to user requests, as well as other information that is not typically available in logs. A packet trace, collected at the Web proxy or server site, can provide important information not available in server logs:

- Timing of packets on the wire
- Out-of-order and lost packets
- Interleaving of packets from different response messages
- TCP-level events such as SYN and FIN packets
- TCP and/or HTTP requests that are not processed by the server
- Amount of data transferred on aborted responses

The value of packet traces has been demonstrated in recent studies on the impact of TCP dynamics on the performance of Web proxies and servers [9, 10]. Similarly, a complete collection of packet traces of both request and response traffic at a Web server would provide a unique opportunity to gauge how a change to HTTP 1.1 would affect the workload.

For example, the packet trace could be used to estimate the latency reductions under persistent connections by measuring the delay involved in closing and reopening a TCP connection between a client and the server for consecutive transfers. As a more complicated example, consider the potential use of range requests in HTTP 1.1 to fetch partial contents of an aborted response message. If a client aborts a request during the transmission of the response, the client (or proxy, if one exists) may receive only a subset of the response. Abort operations can be detected in a packet trace by noting the client RST packet, whereas the server log would either include (or not include) an entry for the request/response. The packet trace would also indicate how much of the transfer completed before the abort reached the server. If the client initiates a second request for the resource, the HTTP 1.0 server would transfer the entire contents again. However, an HTTP 1.1 client (or proxy) could initiate a range request to transfer only the missing portion of the resource. The HTTP 1.0 packet traces would enable us to recognize the client's second request, and model the corresponding range request in HTTP 1.1, assuming the partially-downloaded contents are still in the cache.

4 Implementation

4.1 Server Packet Traces

During the past year and a half, AT&T Labs has built and deployed two high-performance packet monitors at strategic locations inside AT&T WorldNet. Traces from these *PacketScopes* have been used for a number of research studies [10, 11]. For the purposes of this study, we are constructing a third PacketScope to be installed at AT&T's EW3 Web-hosting complex.

This third packet monitor consists of a dedicated 500-MHz Alpha workstation attached to two FDDI rings that together carry all traffic to and from the EW3 server farm. The monitor runs the *tcpdump* utility [12], which has been extended to process HTTP packet headers and keep only the information relevant to our study [13]. The monitor stores the resulting data first to a 10-gigabyte array of striped magnetic disks, then to a 140-gigabyte magnetic tape robot. We ensure that the monitor is passive by running a modified FDDI driver that can receive but not send packets, and by not assigning an IP address to the FDDI interface. We control the monitor by connecting to it over an AT&T-internal network that does not carry customer traffic. We make our traces anonymous by encrypting IP addresses as soon as packets come off the FDDI link, before writing any packet data to stable storage. Our experience with an identical monitor elsewhere in WorldNet indicates that these instruments can capture more than 150 million packets per day with less than 0.3% packet loss.

4.2 Augmented Server Logs

In addition to collecting packet traces, we plan to extend the server logging procedures in EW3 to record additional timing information. A server could log the time it (i) starts processing the client request; (ii) starts writing data into the TCP send socket; and (iii) finishes writing data into the TCP send socket. Typically, servers log just one of the three (often (ii)). But logging all three would allow us to isolate the components of delay at the server. For example, the first two timestamps would allow us to determine the latency in processing client requests (e.g., due to disk I/O, or the generation of dynamic content). The packet traces, coupled with the extended server logs, provide a detailed timeline of the steps involved in satisfying a client request, with limited interference at the server (to log the additional time fields).

4.3 Client Packet Traces

Although our initial study will focus on the server packet traces and the augmented server logs, future work could consider additional measurements at (a limited subset of) the client sites. For example, a packet monitor is already installed at one of the main access points for WorldNet modem customers; this data was used in a recent study of Web proxy caching [10]. This data set would provide a detailed view of the Web traffic for the (admittedly small) subset of EW3 requests that stems from these WorldNet modem customers. By measuring Web transfers at multiple locations, and through multiple measurement techniques, we hope to create a clearer picture of how both the network and the server affect Web performance.

References

[1] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol – HTTP/1.1," September 11 1998.
`ftp://ftp.ietf.org/internet-drafts/draft-ietf-http-v11-spec-rev-05.t%xt`.

[2] AT&T Easy World Wide Web
`http://www.ipservices.att.com/wss/hosting`.

[3] J. C. Mogul, "The case for persistent-connection HTTP," in *Proc. ACM SIGCOMM*, pp. 299–313, August/September 1995.
`http://www.acm.org/sigcomm/sigcomm95/papers/mogul.html`.

[4] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie, and C. Lilley, "Network performance effects of HTTP/1.1, CSS1, and PNG," in *Proc. ACM SIGCOMM*, pp. 155–166, August 1997.
`http://www.inria.fr/rodeo/sigcomm97/program.html`.

[5] R. Caceres, P. Danzig, S. Jamin, and D. Mitzel, "Characteristics of wide-area TCP/IP conversations," in *Proc. ACM SIGCOMM*, pp. 101–112, September 1991.
`http://www.research.att.com/~ramon/papers/sigcomm91.ps.gz`.

[6] K. C. Claffy, H.-W. Braun, and G. C. Polyzos, "A parameterizable methodology for internet traffic flow profiling," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1481–1494, October 1995.
`http://www.nlanr.net/Flowsresearch/Flowspaper/flows.html`.

[7] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *Proc. ACM SIGMETRICS*, June 1998.
`http://cs-www.bu.edu/faculty/crovella/paper-archive/sigm98-surge.ps`.

[8] B. Mah, "An empirical model of HTTP network traffic," in *Proc. IEEE INFOCOM*, April 1997.
`http://www.ca.sandia.gov/~bmah/Papers/Http-Infocom.ps`.

[9] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, M. Stemm, and R. H. Katz, "TCP behavior of a busy Internet server: Analysis and improvements," in *Proc. IEEE INFOCOM*, April 1998.
`http://http.cs.berkeley.edu/~padmanab/index.html`.

[10] R. Caceres, F. Douglis, A. Feldmann, G. Glass, and M. Rabinovich, "Web proxy caching: The devil is in the details," in *Proc. ACM SIGMETRICS Workshop on Internet Server Performance*, June 1998.
`http://www.cs.wisc.edu/~cao/WISP98.html`.

[11] A. Feldmann, A. Gilbert, and W. Willinger, "Data networks as cascades: Explaining the multifractal nature of internet WAN traffic," in *Proc. ACM SIGCOMM*, pp. 42–55, September 1998.

[12] V. Jacobson, C. Leres, and S. McCanne, "`tcpdump`," June 1989.
`ftp://ftp.ee.lbl.gov`.

[13] A. Feldmann, "Continuous online extraction of HTTP traces from packet traces," October 1998. In submission to the W3C Workload Characterization Workshop.